Основы научных расчетов на языке программирования Python

Числа

- Целые числа (int)

```
1, 8, -72
3 847 298 893 721 407…
299_792_458
```

- Числа с плавающей точкой (float)

```
1.2, -0.36
```

- Комплексные числа (complex)

```
1.2 + 3.5j
complex(1.2, 3.5)
```

Числа с плавающей точкой (float)

1.2, -0.36 и 1.67263*10^7

4/3 = 1.33333333333333325951846502...
1/10 = 0.10000000000000000555111512...

$1.67263e^{-7}$ **=** 167263×10^{-7}

Комплексные числа (complex)

```
>>> complex(1.2, 3.5)
(1.2 + 3.5j)
>>>(1.2 + 3.5j).real          #Действительная часть
1.2
>>>(1.2 + 3.5j).imag          #Мнимая часть
3.5
>>>(1.2 + 3.5j).conjugate()  #Сопряженное
(1.2 - 3.5j)
```

Математические функции

```
>>> abs(-5.2)
5.2
>>> abs(3+4j)                    #Норма метрики
5.0
>>> round(-9.62)                 #Округление
-10
>>> round(4.5)
4
>>> round(7.5)
8
```

Модуль math/cmath

```
>>> import math
>>> math.exp(-1.5)          #Экспонента в степени
0.2231301601484298 2
>>> math.cos(0)             #Косинус (в радианах)
1.0
>>> math.sqrt(16)           #Корень числа
4.0
```

`math.sqrt(x)`	$\sqrt{x}$
`math.exp(x)`	e^x
`math.log(x)`	$\ln x$
`math.log(x, b)`	$\log_b x$
`math.log10(x)`	$\log_{10} x$
`math.sin(x)`	$\sin(x)$
`math.cos(x)`	$\cos(x)$
`math.tan(x)`	$\tan(x)$
`math.asin(x)`	$\arcsin(x)$
`math.acos(x)`	$\arccos(x)$
`math.atan(x)`	$\arctan(x)$
`math.sinh(x)`	$\sinh(x)$
`math.cosh(x)`	$\cosh(x)$
`math.tanh(x)`	$\tanh(x)$
`math.asinh(x)`	$\operatorname{arsinh}(x)$

https://docs.python.org/3/library/math.html

`math.factorial(x)`	Факториал x!
`math.erf(x)`	Функция ошибок по x
`math.gamma(x)`	Гамма-функция по x, $\Gamma(x)$
`math.degrees(x)`	Преобразование x из радианов в градусы
`math.radians(x)`	Преобразование x из градусов в радианы

Пример решения:

32.2.4. Всемирная геодезическая система (сеть) (World Geodetic System) – это комплект международных стандартов для описания формы Земли. В самой последней версии WGS-84 земной геоид приближенно определяется как эллипсоид, принимающий форму сжатого у полюсов сфероида с главной, или большой, полуосью эллипса a = 6 378 137.0 м и малой полуосью эллипса c = 6 356 752.314245 м.

Использовать формулу вычисления площади поверхности сжатого у полюсов сфероида

$$S_{obl} = 2\pi a^2(1 + ((1 - e^2)/e)\text{atanh}(e)), \text{ где } e^2 = 1 - (c^2/a^2),$$

для вычисления площади поверхности вышеописанного эллипсоида и сравнить полученный результат с площадью поверхности Земли при предположении, что Земля – сфера с радиусом 6371 км.

```
[18] import math
     a=6_378_137.0
     c=6_356_752.314245
     a,c
```

$$(6378137.0,\ 6356752.314245)$$

```
[11] e = math.sqrt(1-((c**2)/(a**2)))
     e
```

$$0.0818191908429656$$

```
[19] S = 2*math.pi*a**2*(1+((1-e**2)/e)*math.atanh(e)) / 1000000 #в км^2
     S
```

$$510065621.724079$$

NumPy


```
>>> import numpy as np
5.2
>>> a = np.array((100,101,102,103))
>>> a
array([100, 101, 102, 103])
>>> b = np.array( [[1.,2.],[3.,4.]] )
>>> b
array([[1.,2.],
       [3.,4.]])
```

NumPy

```
>>> np.zeros((3,2))     #default dtype = 'float'
array([[0.,0.],
       [0.,0.],
       [0.,0.]])
>>> np.ones((3,3), dtype = int)
array([[1,1,1],
       [1,1,1],
       [1,1,1]])
```

Операции с массивами

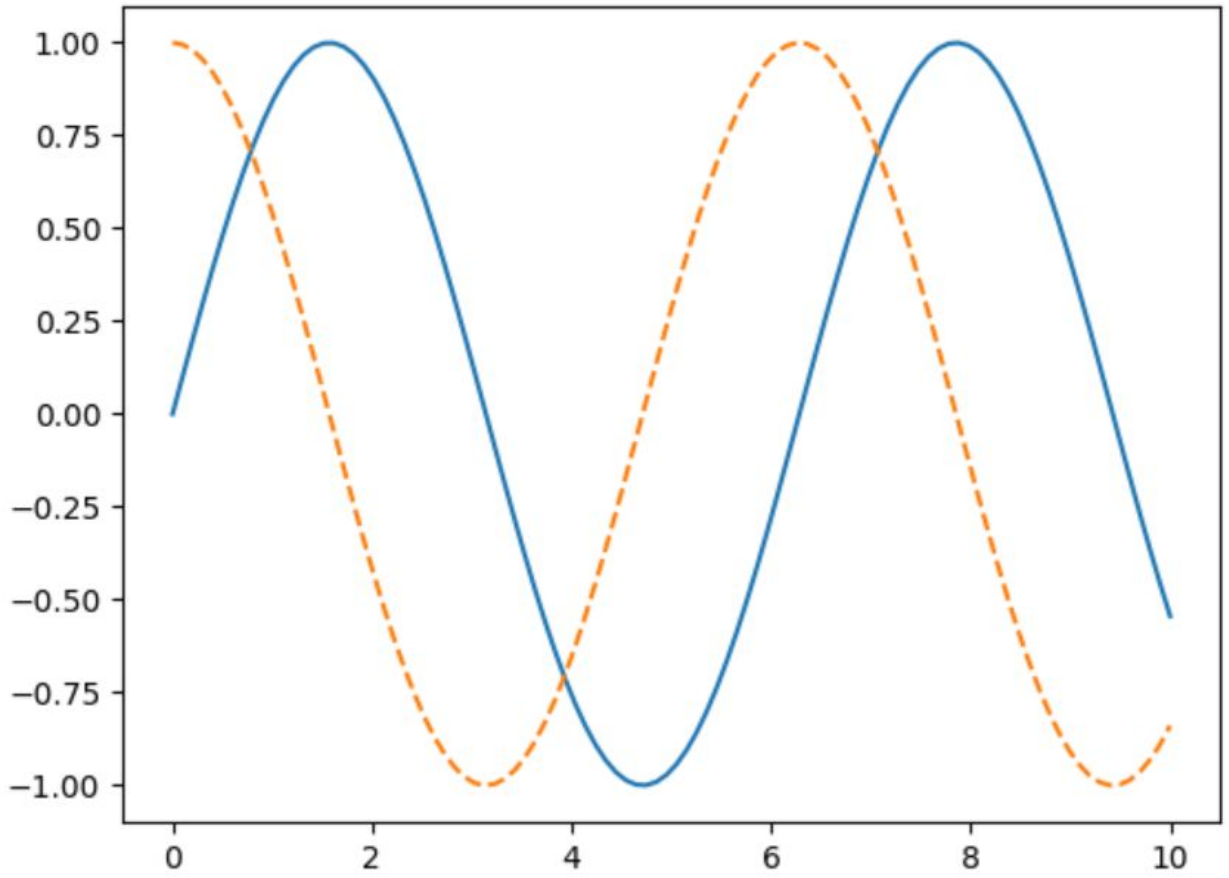
```
>>> A = np.array( [[0,0.5],[-1,2]] )
>>> A*5
array([[ 0.,  2.5],
       [-5., 10. ]])
>>> B = np.array( [[2,-0.5],[3,1.5]] )
>>> A.dot(B)
# np.dot(A,B):скалярное произведение матриц
array([[ 1.5, 0.75],
       [ 4. , 3.5 ]])
```

Операции с массивами

```
>>> A*B                     #Поэлементное умножение
array([[ 0., -2.5],
       [-3.,  3. ]])

>>> A.transpose()           #Или A.T
array([[ 0. ,  0.5],
       [-1. ,  2. ]])
```

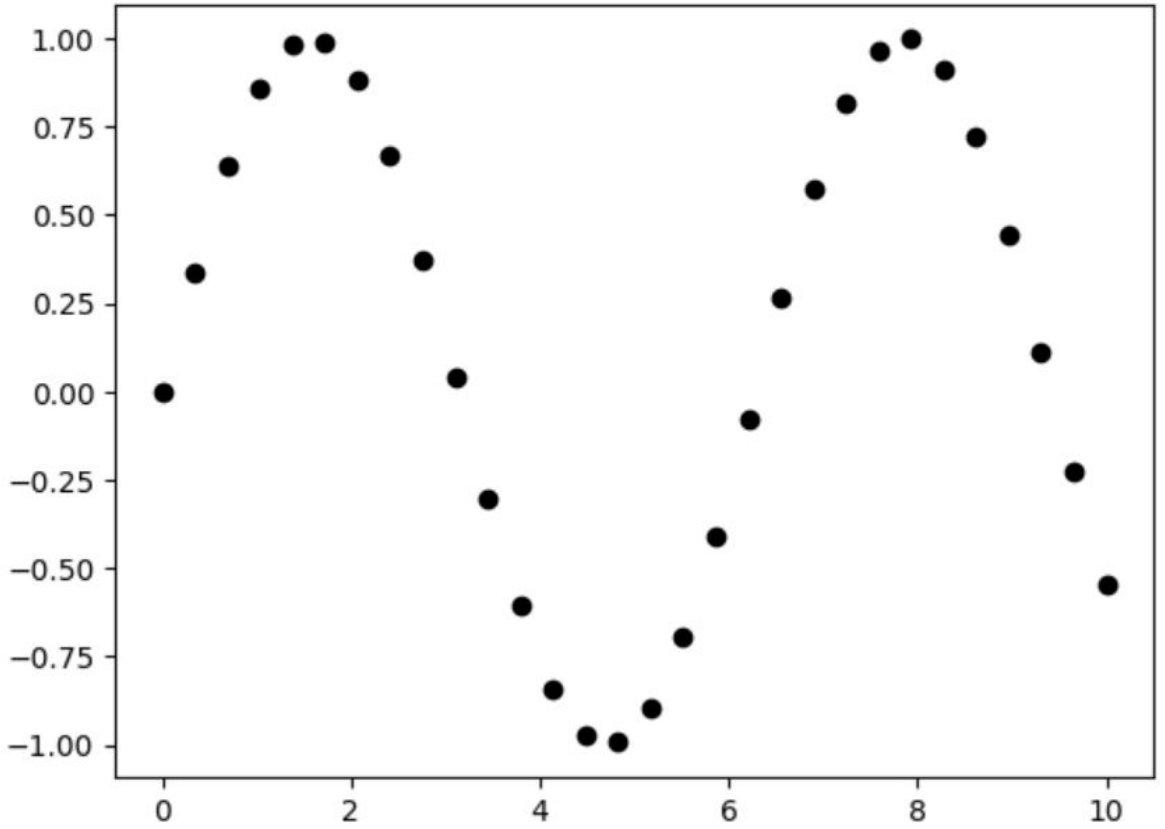
MatPlotLib

```
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(0, 10, 100)

fig = plt.figure()
plt.plot(x,np.sin(x),'-')
plt.plot(x,np.cos(x),'--')
```



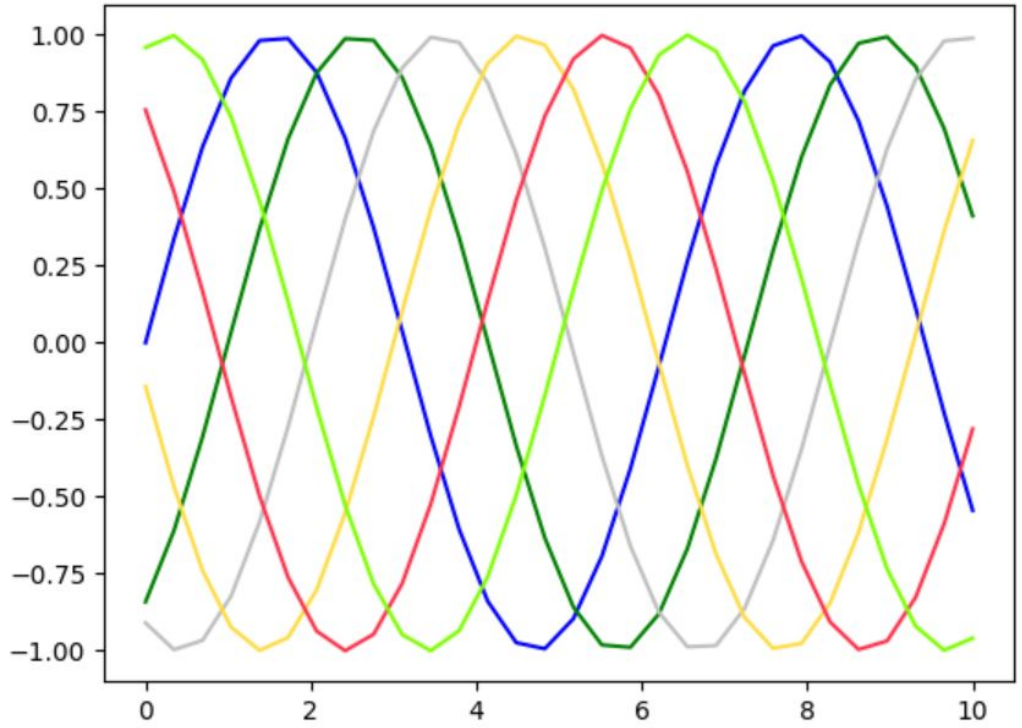
MatPlotLib

```
x = np.linspace(0, 10, 30)
y = np.sin(x)

plt.plot(x,y,'o',
         color = 'black')
```



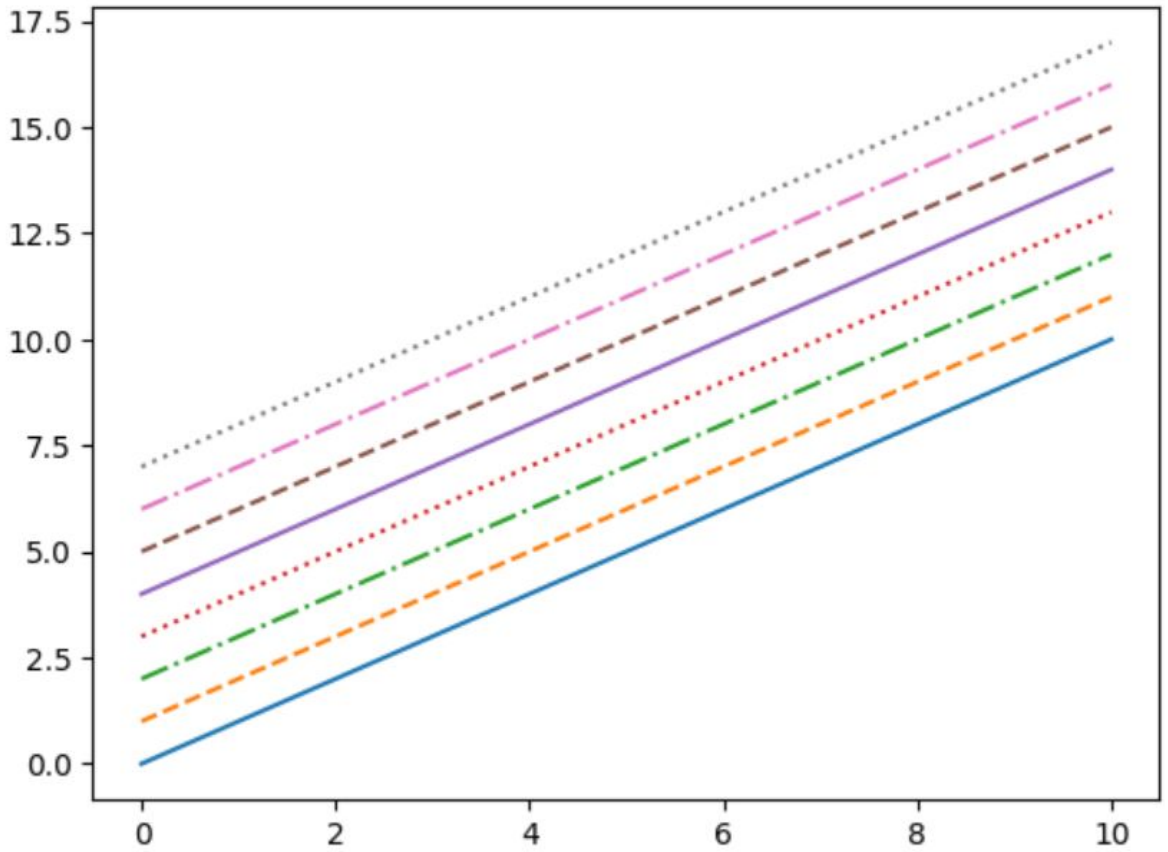
Цвета линий:

```
#Цвет по названию
plt.plot(x,np.sin(x-0), color = 'blue')
#Краткий код цвета
plt.plot(x,np.sin(x-1), color = 'g')
#Шкала оттенков серого
plt.plot(x,np.sin(x-2), color = '0.75')
#RGB(16-ричный)
plt.plot(x,np.sin(x-3), color = '#FFDD44')
#Кортеж RGB
plt.plot(x,np.sin(x-4), color =(1.,0.2,0.3))
#Цвета HTML
plt.plot(x,np.sin(x-5), color =
'chartreuse')
```



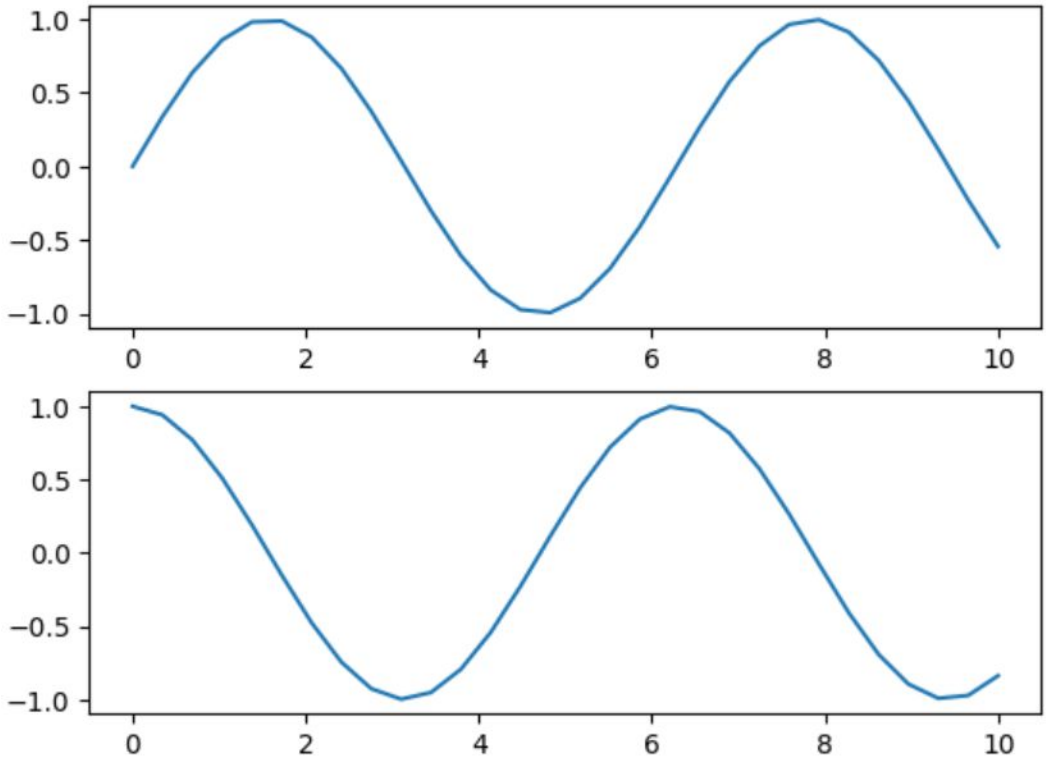
Стили линий:

```
plt.plot(x, x + 0, linestyle='solid')
plt.plot(x, x + 1, linestyle='dashed')
plt.plot(x, x + 2, linestyle='dashdot')
plt.plot(x, x + 3, linestyle='dotted')
# Можно использовать и следующие:

# сплошная линия
plt.plot(x, x + 4, linestyle='-’)
# штриховая линия
plt.plot(x, x + 5, linestyle='--’)
# штрихпунктирная линия
plt.plot(x, x + 6, linestyle='-.’)
# пунктирная линия
plt.plot(x, x + 7, linestyle=':')
```



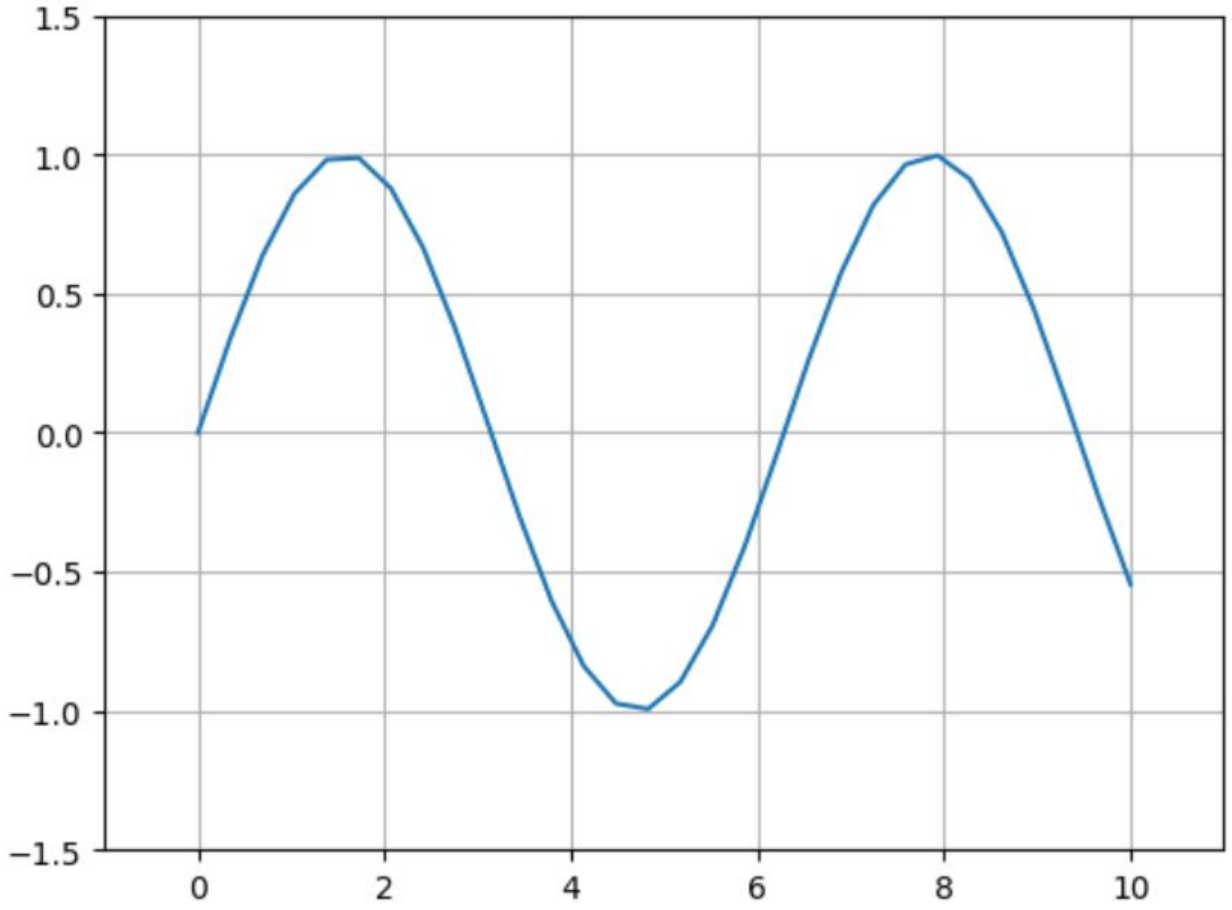
Два рисунка в одном:

```
plt.figure()
#Создём 1-ю область графика и ось
plt.subplot(2,1,1)
#(rows, columns, panel_number)
plt.plot(x,np.sin(x))

#Создаём 2-ю область и ось
plt.subplot(2,1,2)
plt.plot(x,np.cos(x))
```



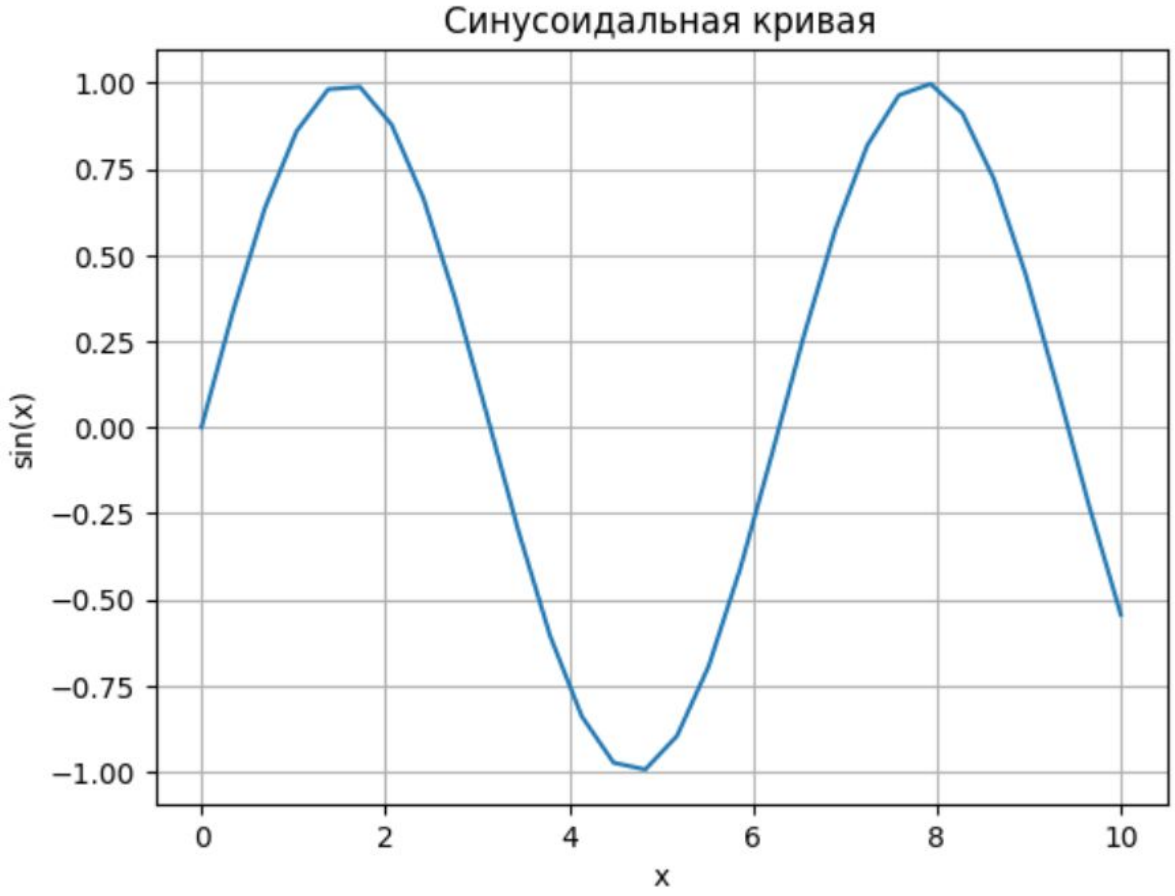
Пределы осей координат:

```
plt.plot(x,np.sin(x))
plt.grid ( True )

plt.xlim(-1, 11)
plt.ylim(-1.5, 1.5)
```



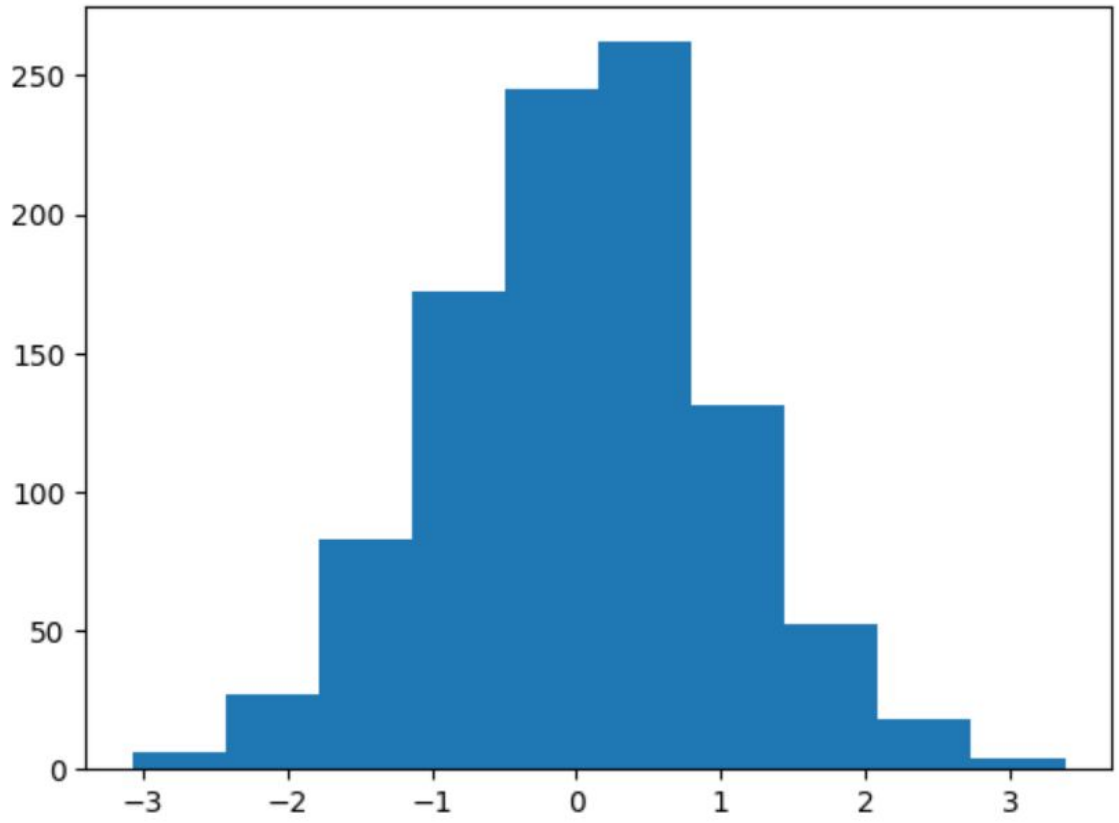
Метки на графиках:

```
plt.plot(x,np.sin(x))
plt.title(“A Sine Curve”)

plt.xlabel(“x”)
plt.ylabel(“sin(x)”)
```



Гистограммы:

```
data = np.random.randn(1000)

plt.hist(data)
```




```
#Импорт модуля integrate
>>>from scipy import integrate
```

constants
special
integrate
optimize
linalg
sparse
interpolate
fftpack
signal
stats

Интегрирование в SciPy

$$\int_0^4 3x^2\,dx$$

```
>>>def f(x):
    return 3.0 * x**2

>>>integrate.quad(f, 0.0, 4.0)
(64.0, 7.105427357601002e-13)
```

Дифференцирование в SciPy

$(3x^2)'$ в точке $x_0 = 2$

```
>>> from scipy import misc
>>>def f(x):
    return 3.0 * x**2

>>> misc.derivative(f,2)
12.0
```

Дифференциальное уравнение в SciPy

$$\frac{dy}{dx} = -2y$$
$$y(0) = 1$$

```
>>>def f(y, x):
    return -2.0 * y
>>>xi = np.linspace(0,1, 10)
>>>y0 = 1.0
>>>sol = integrate.odeint(f, y0, xi)
>>>sol
array([[1.        ], [0.80073742],[0.64118042],[0.51341714],
       [0.41111231],[0.329193  ],[0.26359714],[0.21107209],
       [0.16901331],[0.13533527]])
```

SymPy


```
>>> from sympy import *
>>> init_printing()

>>> x=Symbol('x')
>>> y = x**2 + 3*x - 10
>>> y
```

$$x^2 + 3x - 10$$

Многочлены в SymPy

```
>>> a=(x+y-z)**6
>>> a
```

$$(x + y - z)^6$$

```
>>> a=expand(a) #Раскрытие скобок
>>> a
```

$$x^6 + 6x^5y - 6x^5z + 15x^4y^2 - 30x^4yz + 15x^4z^2 + 20x^3y^3 - 60x^3y^2z + 60x^3yz^2 - 20x^3z^3 + 15x^2y^4 - 60x^2y^3z + 90x^2y^2z^2 - 60x^2yz^3 + 15x^2z^4 + 6xy^5 - 30xy^4z + 60xy^3z^2 - 60xy^2z^3 + 30xyz^4 - 6xz^5 + y^6 - 6y^5z + 15y^4z^2 - 20y^3z^3 + 15y^2z^4 - 6yz^5 + z^6$$

Многочлены в SymPy

```
>>> degree(a,x)   #Степень многочлена
6
>>> collect(a,x) #Собрать со степенями x
```

$$x^6 + 6x^5y - 6x^5z + 15x^4y^2 - 30x^4yz + 15x^4z^2 + 20x^3y^3 - 60x^3y^2z + 60x^3yz^2 - 20x^3z^3 + 15x^2y^4 - 60x^2y^3z + 90x^2y^2z^2 - 60x^2yz^3 + 15x^2z^4 + 6xy^5 - 30xy^4z + 60xy^3z^2 - 60xy^2z^3 + 30xyz^4 - 6xz^5 + y^6 - 6y^5z + 15y^4z^2 - 20y^3z^3 + 15y^2z^4 - 6yz^5 + z^6$$

```
>>> a=factor(a)   #Произведение многочленов
>>> a
```

$$(x + y - z)^6$$

Многочлены в SymPy

```
>>> a = (x**3-y**3)/(x**2-y**2)
>>> a
```

$$\frac{x^3 - y^3}{x^2 - y^2}$$

```
>>> cancel(a) #Сокращение многочлена
```

$$\frac{x^2 + xy + y^2}{x + y}$$

Решение уравнений в SymPy

```
>>> solve(a*x+b,x) #Решение уравнения
```

$$\left[-\frac{b}{a}\right]$$

```
>>> solve([a*x+b*y-e,c*x+d*y-f],[x,y])
```

$$\left\{x : \frac{-bf+de}{ad-bc}, \quad y : \frac{af-ce}{ad-bc}\right\}$$

```
>>> roots(x**3-3*x+2,x) #Корни многочлена
```

$$\{-2 : 1, \quad 1 : 2\}$$

Дифференцирование и интегрирование в SymPy

```
>>> y = x**2 + 3*x - 10
>>> diff(y)
```

$$2x + 3$$

```
>>> y = x**2 + 3*x - 10
>>> integrate(y)
```

$$\frac{x^3}{3} + \frac{3x^2}{2} - 10x$$

Разложение на множители в SymPy

```
>>> n = 1293784932423423423482349623642438
>>> factorint(n)
```

$$\{2 : 1, 23 : 1, 113 : 1, 821 : 1, 7178823570343 : 1, 42230806618727 : 1\}$$

Математические константы

```
>>> c = pi*sqrt(2)*exp(1)
>>> c
```

$$\sqrt{2}e\pi$$

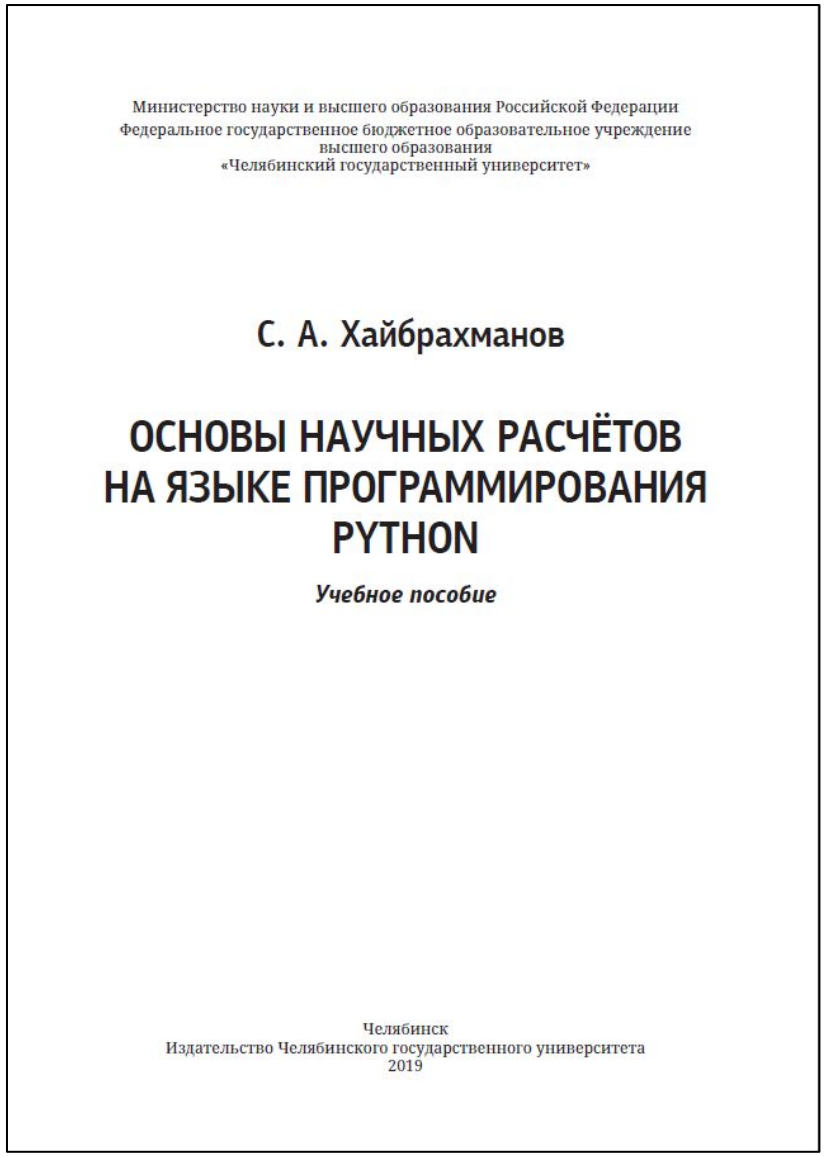
```
>>> c.evalf()
```

$$12.07700795676666$$

```
>>> cel_num = float(c.evalf())
>>> cel_num
12.07700795676666
```

Книги

- «Основы научных расчетов на языке программирования Python» учеб. пособие / С. А. Хайбрахманов. — Челябинск : Изд-во Челяб. гос. ун-та, 2019. — 96 с.



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Челябинский государственный университет»

С. А. Хайбрахманов

ОСНОВЫ НАУЧНЫХ РАСЧЁТОВ
НА ЯЗЫКЕ ПРОГРАММИРОВАНИЯ
PYTHON

Учебное пособие

Челябинск
Издательство Челябинского государственного университета
2019

Книги

- Плас Дж. Вандер Python для сложных задач: наука о данных и машинное обучение. — СПб.: Питер, 2018. — 576 с.

Книги

- Кристиан Хилл Научное программирование на Python 2021. – 646 с.